日本語構文解析システム
KNP version 3.0
使用説明書

黒橋 禎夫　　河原 大輔

平成 21 年 9 月

KN parser (Kurohashi-Nagao parser) 3.0 Users Manual

Version 0.9	1 June 1993
Version 1.0	24 August 1994
Version 2.0b1	29 November 1996
Version 2.0b2	3 December 1996
Version 2.0b3	25 June 1997
Version 2.0b4	5 September 1997
Version 2.0b5	29 October 1997
Version 2.0b6	9 June 1998
Version 2.0	7 September 2005
Version 3.0	30 September 2009

目 次

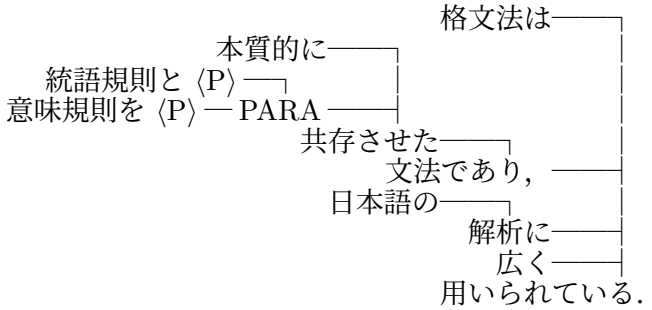
1 KNP とは

KNP は日本語文の構文解析を行うシステムです．形態素解析システムの解析結果 (形態素列) を入力とし，それらを基本句 (後述) および文節単位にまとめ，それらの間の係り受け関係を決定します．また，自動獲得した格フレームに基づき、格関係の解析も行います．

以下に典型的な使用例を示します．

```
% cat test
格文法は本質的に統語規則と意味規則を共存させた文法であり，日本語
の解析に広く用いられている．
```

```
% juman -e2 -B < test | knp
```



このシステムの特徴は次のような点にあります．

- 句の並列 (「〜と〜」)，節の並列 (「〜し，〜する」) などを文節の類似性を調べて取り出す機能，例外的な文節の振る舞いを個別に記述できる機能などをもっている．
- これらの機能を用いて，一意の解析結果を (できるだけ正しく) 求めるという立場をとっている．現在のところ複数解を出力するオプションはない (改良の予定)．
- 基本的には，このシステムの上でユーザが独自の文法を構築することは想定していない．しかし，ユーザによるある程度の文法修正は可能である．

京都大学黒橋研究室では現在もシステムの改良を継続的に行っており，ユーザの皆様からの問題点の指摘を歓迎致します (連絡先：nl-resource@nlp.kuee.kyoto-u.ac.jp)．

2 インストール方法

KNP は以下のツール・辞書を用いますので，あらかじめご用意ください．

1. JUMAN 6.0 以上
   (http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html)
2. 分類語彙表 増補改訂版
   (http://www.kokken.go.jp/katsudo/kanko/data/index.html)

3. Perl
   (http://www.perl.com/)

1 は，テキストを形態素列に変換するプログラムで，KNP はこのプログラムの出力を入力とします．2 は並列構造解析で利用しますが，なくても動作します (解析精度は若干低下します)．3 は分類語彙表をインストールする場合とルールファイルを更新する場合に必要となります．

2.1 Unix 系 OS におけるインストール方法

KNP のインストールは以下の手順で行います．

1. knp-3.0.tar.gz を展開し，knp-3.0 ディレクトリに移動する．
2. ‘./configure’ を実行する．
   ‘./configure’ にオプションを与えることにより，インストール先やコンパイルオプションなどを変更することができる．分類語彙表を用いる場合には，

   ```
   --with-bgh-file=/somewhere/bunruidb.txt
   ```

   のようなオプションによって，分類語彙表ファイル (bunruidb.txt) の場所を指定する．詳細は，同梱の INSTALL または ‘./configure --help’ の出力を参照のこと．
3. ‘make’ を実行する．
   KNP システムのコンパイルと辞書の構築が行われる．
4. (root で)‘make install’ を実行する．
   KNP システムと辞書がインストールされる．インストールされる場所は以下のとおりである．$PREFIX は，デフォルトでは/usr/local であるが，‘./configure --prefix’ で設定できる．

$PREFIX/bin/knp	実行ファイル
$PREFIX/share/knp/dic/	辞書
$PREFIX/share/knp/rule/	ルール
$PREFIX/etc/knprc	設定ファイル
$PREFIX/share/knp/doc/manual.pdf	マニュアル
$PREFIX/libexec/knp/	辞書作成プログラム

2.2 Windows におけるインストール方法

knp-3.0.exe を実行し，表示される指示に従ってインストールを行います．以下のファイルがインストールされます．$PREFIX は，デフォルトでは C:¥Program Files¥knp ですが，インストール途中で設定できます．

$PREFIX¥knp.exe	実行ファイル
$PREFIX¥dic	辞書
$PREFIX¥rule	ルール
$PREFIX¥manual.pdf	マニュアル

分類語彙表を利用するには，KNP をインストールした後に，以下の手順を実行する必要があります.

1. 分類語彙表 CD-ROM に含まれている bunruidb.txt を$PREFIX¥dic¥scode¥bgh にコピーする.
2. $PREFIX¥dic¥scode¥bgh¥make_bgh_db.bat を実行する.

Windows 版は UNIX 版と以下の点が異なります.

- 入出力は SJIS で行う.
- サーバ・クライアント機能は備えていない.

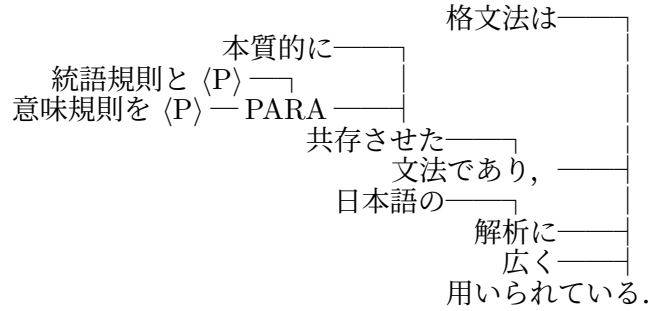
3 利用方法

3.1 コマンドラインからの利用方法

KNP の入力は，形態素解析システム JUMAN の解析結果です．以下に典型的な利用方法を示します.

```
% cat test
格文法は本質的に統語規則と意味規則を共存させた文法であり，日本語
の解析に広く用いられている.
```

```
% juman -e2 -B < test | knp
```



KNP では,「基本句」を単位にして解析しています。基本句とは，京都大学テキストコーパスに含まれる格関係，照応・省略関係，共参照タグ付きコーパスにおいて各種関係を付与している単位です．基本句は，基本的に自立語一語を核として，その前後に存在する接頭辞，接尾辞，助詞，助動詞などの付属語をまとめたもので，文節と同じかその一部です.

現在の KNP では，文節内の基本句間の係り受けは扱っておらず隣に係るように解析されます.

KNP には以下のオプションがあります.

- 解析のレベルの指定

-bnst	形態素列を文節列に変換する
-dpnd	さらに，文節間の係り受け解析を行う
-case (default)	さらに，格関係の解析を行う

- 解析結果の表示の指定

-tab	表形式による表示
-tree (default)	木構造による表示
-bnsttab	文節の表形式による表示
-bnsttree	文節の木構造による表示
-sexp	リスト形式による表示

- 解析結果の出力情報の指定

-normal (default)	最終的解析結果だけを表示
-detail	係り受けの可能性行例，文節間の類似度行列なども表示
-debug	さらに詳しい解析途中段階の情報の表示

3.2 Perl からの利用方法

Perl モジュール “KNP” を用いることにより，Perl から KNP を利用することができます.

- インストール方法

  KNP を展開したディレクトリにある `perl` ディレクトリに移動し，次の手順を実行します.

  1. perl Makefile.PL
  2. make
  3. (root で) make install

- 使用例

  プログラムの例を以下に示します．このプログラムを実行すると,「この文を形態素解析してください.」という文の解析結果が表示されます.

```
use KNP;
$knp = new KNP;
$result = $knp->parse( "この文を構文解析してください．" );
print $result->all();
```

  詳細は ‘`perldoc KNP`’ の実行結果を参照してください.

4 解析過程の概要

KNP は，JUMAN の出力形態素列 (同形異義語を含む) を入力とし，以下のような処理を行います.

1. 同形異義語を処理し，一意の形態素列に変換します (次節参照:mrph_homo.rule).
2. 各形態素に対して，その働きを示す種々のマーク (feature) を与えます．辞書的情報，文節にまとめるための自立語・付属語の例外的情報などです (mrph_basic.rule).
3. 形態素に与えられた feature にしたがって，形態素列を文節列に変換します.
4. 各文節に対して，その働きを示す feature を与えます．用言，体言，ガ格，ヲ格，並列構造の可能性などの feature です (bnst_basic.rule, bnst_type.phrase, bnst_etc.rule).
5. 文中に並列構造の可能性を示す表現があれば，その前後で類似する文節列を検出し，それを並列構造の範囲とします.
6. (並列構造がある場合は，その検出された範囲と矛盾しないように) 文全体の係り受け構造の候補をつくり出します (kakari_uke.rule).
7. つくり出された各候補を評価し，もっとも優先的なものを解として出力します．評価は，自動獲得された格フレームに基づき，述語項構造を評価することによって行います.

5 文法の記述

文法を記述するファイルの形式には，phrase, rule, data の 3 種類あります．人が記述するのは phrase または rule 形式で，KNP は data 形式を読み込みます．phrase 形式のファイルは phrase2rule.pl プログラムを用いて rule 形式に変換し，rule 形式は rule2data.pl プログラムを用いて data 形式に変換します[1].

phrase 形式と rule 形式の違いはルールの書きやすさで，phrase 形式の方が人にとって書きやすい書式となっています．rule 形式のファイルはリスト形式で記述しますが，そのルールの使用目的によって書式が異なり，同形異義語，形態素，文節，係り受けの 4 種類があります．文節ルールは，基本句に対しても適用されます．KNP が用いているルールファイルは以下のとおりです.

[1] これらの変換は rule ディレクトリで 'make' を実行することによって行われます．なお配布パッケージには data 形式ファイルも含まれているので，文法の変更を行なわない場合は何もする必要はありません.

	ファイル形式	ルール書式
mrph_homo.rule	rule	同形異義語
mrph_filter.rule	rule	形態素
mrph_auto_dic.rule	rule	形態素
mrph_basic.rule	rule	形態素
mrph_ne.rule	rule	形態素
bnst_basic.rule	rule	文節
bnst_type.phrase	phrase	(文節)
bnst_etc.rule	rule	文節
case_analysis.rule	rule	文節
kakari_uke.rule	rule	係り受け

まず，rule 形式のルール記述に用いられる feature パターン，形態素パターン，文節パターンについて説明します．次に，同形異義語，形態素，文節，係り受けの 4 つのルール書式について説明します．最後に phrase 形式のルール記述について説明します．

5.1 feature パターン

形態素，文節には feature の集合が与えられます．たとえば「彼は」という文節には

```
{<体言> <提題> <係:未格>}
```

というような feature 集合が与えられます[2]．

feature パターンはこのような feature 集合に対するパターンで，feature の積和標準形で与えます．また feature 名の直前に ^ をつければ feature の否定，すなわちその feature が feature 集合に含まれないということを指定できます．たとえば

```
((体言 提題)(体言 ^時間))
```

は，〈体言〉，〈提題〉という feature をともに含むか，あるいは〈体言〉という feature を含み〈時間〉という feature を含まない feature 集合とマッチします．

5.2 形態素パターン

形態素パターンは，任意の形態素とマッチすることを示す

```
?
```

あるいは，具体的に

```
[ 品詞 品詞細分類 活用型 活用形 語彙 feature パターン ]
```

[2]feature の中に T で始まるものがありますが，これはルール記述をコンパクトにするために一時的に与える feature で，解析結果の出力には表示されません．

という形で指定します．[···] の直前に ^ を付与することで，具体的に指定した形態素以外のもの，というパターンを表現することもできます．品詞，細分類，活用型，活用形，語彙はそれぞれ

具体的指定 — 名詞，など
具体的指定のリスト — (名詞 動詞) など
∗ (任意であることを示す)

のいずれかで (品詞体系などは JUMAN に従う)，この場合にも直前に ^ を付加することで具体的に指定したもの以外とマッチすることを指定できます．feature パターンは上で説明した feature の積和標準形で，形態素に与えられている feature 集合に対する条件となります．形態素パターンにおいて，ある部分から後ろがすべて ∗ で，feature パターンの条件もない場合は，それらを省略することができます．

例えば以下のような指定が可能です．

```
[ 動詞 * * * ( する できる ) ] ; 動詞「する」または動詞「できる」
^[ 動詞 * * * ( する できる ) ] ; 動詞「する」，動詞「できる」以外
[ 動詞 * * * ^( する できる ) ] ; 「する」,「できる」以外の動詞
[ (名詞 副詞) * * * * ((漢字)) ] ; <漢字>という feature をもつ名詞または副詞
[ 動詞 *(サ変動詞 カ変動詞) ] ; 動詞で，活用型がサ変動詞またはカ変動詞のもの
```

さらに，具体的な形態素指定または ? の直後に ∗ を付加すると，それらの 0 回以上の繰り返しとマッチ可能であることを示します．例えば，

```
?*
```

は任意の形態素列と，

```
[ 助詞 ]*
```

は任意の助詞列とマッチします．(注: ∗ は形態素内部の指定では任意のものとのマッチを，外部では条件の繰り返しを指定し，働きが異なる．)

5.3 文節パターン

文節パターンは，任意の文節とマッチすることを示す

```
?
```

あるいは，具体的に

```
<(形態素パターン列) feature パターン >
```

という形で指定します．形態素パターン列は文節を構成する形態素列に対するパターンで，feature パターンは文節がもつ feature 集合に対するパターンです．さらに，具体的な文節指定または ? の直後に ∗ を付加すると，それらの 0 回以上の繰り返しとマッチ可能であることを示します．

たとえば，

```
< (?* [助詞 * * * と] [助詞 * * * も]* [特殊 読点]* ) ((用言)) >
```

というパターンは,「〜と」,「〜とも,」などの形態素列で,〈用言〉という feature を持つ文節にマッチします.

なお,形態素パターンを囲む括弧は [ ],文節パターンを囲む括弧は< >としていますが,これはルールの読みやすさのためのもので,KNP はこれらを ( ) に変換した data 形式のファイルを参照します.

5.4 同形異義語ルール

同形異義語は,mrph_homo.rule に与えられた次の形式のルールによって処理されます.

((形態素パターン列) (形態素パターン $_1$ 形態素パターン $_2$ $\cdots$ 形態素パターン $_n$) (形態素パターン列) feature 列)

最初の要素は処理対象としている形態素の左側の形態素列に対するパターン,2 番目の要素は,処理対象としている形態素に対する同形異義語パターンで,それらと 1 対 1 対応のマッチングがとれる場合に適用されます.3 番目の要素は処理対象としている形態素の右側の形態素列に対するパターンです.この規則が適用されると,同形異義語の中から先頭の形態素パターン $_1$ にマッチした形態素が選択され,さらに,その形態素に feature 列が与えられます.

たとえば,

```
( ( ?* ) ( [動詞] [連体詞] ) 弱動詞 )
```

というルールは「ある」などの動詞と連体詞の同形異義語に適用され,動詞の解釈を優先し,それに〈弱動詞〉という feature を与えるという働きを持ちます.

5.5 形態素ルール

形態素に対する feature は次のような形のルールによって付与されます.

```
( ( 形態素パターン列 ) ( 形態素パターン ) ( 形態素パターン列 ) feature 列 )
```

最初の要素は処理対象としている形態素の左側の形態素列に対するパターン,2 番目の要素は処理対象としている形態素自身に対するパターン,3 番目の要素は処理対象としている形態素の右側の形態素列に対するパターンです.これら全ての条件にマッチした場合,その形態素に feature 列が与えられます.

たとえば,

```
( ( [* * * 意志形] [助詞 * * * と] ) ( [動詞 * * * する] ) ( ) 付属 )
```

というルールは,左側に「しよう＋と」などの形態素列がある場合の動詞「する」に〈付属〉という feature を与えるためのもです.

このようなルールは mrph_basic.rule というファイルにまとめられており,入力の全形態素に対して,ファイル中の全ルールの適用を試みます.

5.6 文節ルール

文節に対する feature は次のような形のルールによって付与されます.

```
( ( 文節パターン列 ) ( 文節パターン ) ( 文節パターン列 ) feature 列 )
```

最初の要素は処理対象としている文節の左側の文節列に対するパターン，2 番目の要素は処理対象としている文節自身に対するパターン，3 番目の要素は処理対象としている文節の右側の文節列に対するパターンです．これら全ての条件にマッチした場合，その文節に feature 列が与えられます.

たとえば,

```
( ( )
  ( <([動詞 * 母音動詞 基本形 できる] [助詞 * * * だけ]) > )
  ( <(?*)((用言)) > )
  弱用 )
```

というルールは，右側に〈用言〉という feature をもつ文節があり，それ自身が「できる＋だけ」という形態素列である文節に〈弱用〉という feature を与えます.

文節の係り受けに関して重要な feature は bnst_type.phrase(5.8 節参照) で与えられています．これら以外に文節の基本的な feature と例外的な feature は bnst_basic.rule, bnst_etc.rule というファイルで与えられます．これらは，bnst_basic.rule, bnst_type.phrase, bnst_etc.rule という順に適用されます．なお，bnst_type.phrase では各文節に対してファイルの先頭から順にルールの適用を試み，いずれかのルールが適用されればその文節に対する処理を終了します．これら以外のルールファイルでは，全文節に対して，ファイル中の全ルールの適用が試みられます.

5.7 係り受けルール

係り受けの規則は文節自身のパターンを指定するのではなく，これまでの規則によって与えられている文節の feature 集合に対して記述します．係り受けの規則は次の形で与えます.

```
( 係り側 feature パターン
    ( [ 受け側 feature パターン 1 係り受けタイプ 1 ]
      [ 受け側 feature パターン 2 係り受けタイプ 2 ]
                ...
      [ 受け側 feature パターン n 係り受けタイプ n ] )
    係り受け停止 feature パターン
    優先係り受け位置 )
```

係り側 feature パターンは係りの文節に対する feature パターン，**受け側 feature パターン** $_i$ は受けの文節に対する feature パターン，**係り受けタイプ** $_i$ は，それらにマッチした係り文節と受けの文節間に与える係り受け関係を表します．係り受け関係としては並列 (P)，同格 (A)，それ以外 (D) の 3 種類があります.

ある文節に対して係り先の文節を探す場合は，その文節と係り側 feature パターンがマッチする係り受け規則を探し，その規則のいずれかの受け側 feature パターンにマッチする文節がないかどうかを，処理文節の直後の文節から順に調べます．この時，**係り受け停止 feature パターン**を「すでに (近くに) 係り得る文節がある場合，そのパターンにマッチする文節 (停止文節) を飛び越えるような係り受け関係は考慮しない」というかたちで用います．係り受け停止 feature パターンを指定しない場合はすべての文節が停止文節になりえる，すなわち係り得る文節が 1 つみつかれば，それ以上他の (遠くの文節に係る) 可能性を考慮しない，ということを意味します．

優先係り受け位置は整数を指定し，1 なら係り得る最も近い文節，2 なら係り得る 2 番目に近い文節，−1 なら係り得る最も遠い文節を優先的な係り先とします．

たとえば，

```
（（（係:デ格 読点）（係:カラ格 読点）（係:マデ格 読点））
  （［（（用言:強））D］）
  （（レベル:C）（レベル:B’））
  2 ）
```

という規則は，〈係:デ格〉と〈読点〉，〈係:カラ格〉と〈読点〉などの feature を持つ文節が〈用言:強〉という feature を持つ文節に D(通常の係り受け関係) で係り，〈レベル:C〉または〈レベル:B’〉という feature を持つ文節があればそれ以上遠くの文節に係る可能性は考慮せず，最優先されるのは係り得る 2 番目に近い文節である，ということを意味します．

5.8 phrase 形式

phrase 形式は，文節ルールを人にとって読み書きやすい形で記述するためのもので，1 行 1 ルールで，次のような書式になっています．

```
[前の文脈] 自分自身 [後の文脈]　　FEATURE 列
```

「前の文脈」は処理対象としている文節の左側の文節列に対するパターン,「自分自身」は処理対象としている文節自身に対するパターン,「後の文脈」は処理対象としている文節の右側の文節列に対するパターンです．これら全ての条件にマッチした場合，対象文節に FEATURE 列が与えられます．パターンと FEATURE 列の間には 1 つ以上のタブを入れる必要があります．

たとえば，

```
書き [^次第‥]　　ID:〜（次第）
```

というルールは，右隣に「次第」という文節があり，それ自身が連用形である文節に〈ID:〜（次第）〉という feature を与えます.「書く」は予約語で動詞に汎化されます．^ は文節の先頭に，‥ は任意の形態素列にマッチします．記述の詳細は phrase2rule.pl の冒頭部分のコメントを参照してください．

phrase 形式のファイルは phrase2rule.pl によって rule 形式に変換されます．この変換は，Perl モジュール KNP.pm を通して juman/knp を用いて行いますので，phrase2rule.pl を

実行するためには，juman，knp，KNP.pm をあらかじめインストールしておく必要があります[3].

参考文献

[1] 黒橋禎夫, 長尾眞. 長い日本語文における並列構造の推定. 情報処理学会論文誌, Vol.33, No.8, pages 1022-1031 (1992.8).

[2] 黒橋禎夫, 長尾眞. 並列構造の検出に基づく長い日本語文の構文解析. 自然言語処理, Vol.1, No.1, pages 35-57 (1994.10).

[3] 黒橋禎夫, 長尾眞. 京都大学テキストコーパス・プロジェクト. 言語処理学会 第 3 回年次大会, pages 115-118 (1997.3).

[4] 黒橋禎夫. 開発されるべきシステムとしての言語. 月刊「言語」, Vol.27, No.6, pages 66-73 (1998.6).

[5] 黒橋禎夫. コーパスが先か，パーサーが先か. 情報処理, Vol.41, No.7, pages 769-773 (2000.7).

[6] 黒橋禎夫. 結構やるな，KNP. 情報処理, Vol.41, No.11, pages 1215-1220 (2000.11).

[7] 河原大輔, 黒橋禎夫. 自動構築した大規模格フレームに基づく構文・格解析の統合的確率モデル. 自然言語処理, Vol.14, No.4, pages 67-81 (2007.7).

[3] ここでは knp は同音異義語ルール，形態素ルールを参照して文節列への変換を行うだけです. phrase2rule.pl による文節ルールの変換との間に相互作用／副作用はありません.